Programación I – Eval1

Nombre: .....Apellidos: .....

Grupo: ..... Nº mat: .....email:..... Grado(II/ADE):.....

Duración de la prueba: 1h 30m

1. Definir en Java una función *sumaImpares* que recibe dos enteros *a* y *b*, y devuelve la suma de todos los números impares contenidos en el intervalo cerrado [a , b].

Ejemplos:
sumaImpares(3, 7) = 3+5+7 = 15
sumaImpares(7, 3) = 0
sumaImpares(4, 8) = 5+7 = 12

(**1 punto**)

```
static int sumaImpares (int a , int b) {
   int resultado  = 0;
   for (int i = a; i <= b; i ++) {
    if(i%2 ¡=0)
      resultado = resultado + i;
   } // fin for
   return resultado;
 } // fin sumaImpares
```

---

2. Redactar en Java tres pruebas para la función *sumaImpares* del apartado anterior
(**1 punto**)

```
boolean  prueba1 = sumaImpares(3, 7) == 15;
boolean  prueba2 = sumaImpares(7, 3) == 0;
boolean  prueba3= sumaImpares(5, 6) == 5;
```

---

3. Definir en Java la función *traducirDireccion* que recibe un carácter representando un punto cardinal en nomenclatura inglesa (*N, S, E, W*) y devuelve la palabra correspondiente en español (*Norte*, *Sur*, *Este*, *Oeste).* En caso de introducir un carácter erróneo, deberá devolver el texto "*DIRECCIÓN ERRONEA*".
(**1 punto**)

Solución:

```
static  String  traducirDireccion (char dir)  {
   switch (dir) {
    case 'N':  return "Norte";
    case 'S':  return "Sur";
    case 'E':  return "Este";
    case 'W':  return "Oeste";
    default: return  "DIRECCIÓN ERRONEA ";
   } // fin switch
  } // fin de  traducirDireccion
```

---

4. Escribir cual será la salida al ejecutar el siguiente código.
(**2 puntos**)

```
class Ejercicio4  {

 static boolean incognita (int uva, int gusano, int pera){
  int escogido = pera;
  boolean horror = false;
  while (escogido <= uva && !horror) {
    if (escogido == gusano)
       horror = true;
    escogido = escogido +1;
  } // de while
 return horror ;
} //de incognita

 public static void main ( String [ ] arg ) {
   System.out.println ("incognita (6,4,2) = " +incognita (6,4,2));
   System.out.println ("incognita (6,14,2) = " +incognita (6,14,2));
 } // fin main
}  // fin Ejercicio4
```

Solución

```
>  incognita (6,4,2) =  true
>  incognita (6,14,2) = false
```

---

5. Definir en Java la función *sonPitagoricos* que recibe tres enteros (*a, b, c)* y comprueba si forman una terna pitagórica.
Se dice que una terna de valores enteros es pitagórica si se cumple: *"el cuadrado de uno de ellos es igual a la suma de los cuadrados de los otros dos"*.

Ejemplos:

sonPitagoricos (4, 3, 5) = true puesto que se cumple que $5^2 = 4^2 + 3^2$
sonPitagoricos (17, 15, 8) = true puesto que se cumple que $17^2 = 8^2 + 15^2$
sonPitagoricos (1, 2, 3) = false puesto que no se cumple la condición del enunciado.

nota: Se puede utilizar, si se considera oportuno, como si estuviesen ya implementadas, las funciones *mayor2(a,b)* y *menor2(a,b)* que, respectivamente, devuelven el mayor o el menor de dos valores enteros.
Igualmente se puede utilizar, como funciones auxiliares ya implementadas, *mayor3(a,b,c)* o *menor3(a,b,c)* , que devuelven, respectivamente, el mayor o el menor de 3 valores enteros.

(**2 puntos**)

Solución1

```
static boolean sonPitagoricos ( int a, int b, int c)  {
   int mayor = mayor2(mayor2(a, b), c);       //  mayor valor  ( hipotenusa)
   int menor = menor2(menor2(a, b), c);      //  menor valor ( un cateto)
   int otro = a+b+c-mayor-menor;     //  otro es un posible cateto
   return   mayor*mayor ==menor*menor+otro*otro;
 }
```

Solución2

```
static boolean sonPitagoricos (int a, int b, int c)  {
  return   a*a == b*b+c*c || b*b == a*a + c*c || c*c = a*a+ b*b;
}
```

Solución3

```
static boolean esTerna (int x, int y, int z)
{
  return x*x == y*y + z*z;
}
static boolean sonPitagoricos (int a, int b, int c)
{
  return esTerna(a,b,c) || esTerna(b,c,a) || esTerna(c,a,b);
}
```

---

6. Dado el siguiente código para la función *misterio* que usa la función auxiliar *esImpar*.

```
static int misterio (int num) {
    int resultado = 0;
    int pos =1;
   int digito;
    for (int numero = Math.abs(num); numero > 0; numero=numero/10) {
        digito = numero%10;
        if(esImpar(pos))
          resultado = resultado + digito;
        pos++;
    } // fin del  for
    return resultado;
} // fin misterio
```

Se pide:

6.1 Dar el resultado de llamar a esa función con los siguientes parámetros de entrada: 3428 y 8. **(0,5 punto)**

---

misterio (3428) = 12;
misterio (8) = 8

6.2 Dar una especificación informal de la función, es decir, una descripción sencilla y completa en español de lo que hace la función *misterio*. **(0,5 punto)**

Calcula la suma de los dígitos de num que están en las posiciones impares. Los dígitos se numeran de derecha a izquierda, siendo la posición 1 la de las unidades, la 2 la de las decenas, y así sucesivamente.

---

7. Declarar en Java una función *sumaSerieAritmetica* que recibe tres enteros *a* , *n* , *d* y devuelve la suma de los *n* primeros términos de una serie aritmética donde *a* es el primer valor de la serie y *d* es la distancia entre dos términos consecutivos de la serie.

Nota: No utilizar la expresión de la suma de una serie aritmética.

Ejemplos:

sumaSerieAritmetica (1, 4, 1) = 1+2+3+4 = 10
sumaSerieAritmetica (1, 3, 3) = 1+4+7 = 12

(**2 puntos**)

Solución

```
static int sumaSerieAritmetica ( int a , int n , int d ) {
    int suma = 0;
    for ( int i = 1 ; i <= n; i++ ) {
        suma = suma + (a+d*(i-1));
    }
    return suma;
}
```